PSCD

TRABAJO FINAL
2017/2018
DISEÑO

DANIEL NAVAL ALCALÁ
VÍCTOR MIGUEL PEÑASCO ESTÍVALEZ
PABLO ORDUNA LAGARMA

Índice

Introducción

El sistema de vallas electrónicas publicitarias implementado se basa en un modelo cliente servidor distribuido, en el que el servidor es el principal protagonista ya que se encargará de establecer y vender una valla publicitaria durante una cantidad de tiempo, mostrar por pantalla la información que se encuentra gestionando con otro proceso al que le llamaremos estadístico y controlar las vallas de los pujadores ganadores.

El cliente se conectará en el momento que tiene disponible para empezar la subasta y en ese momento empezarán las pujas para obtener el servicio publicitario.

Cuando un cliente gane una subasta, éste enviará la imagen que querrá mostrar y empezará una subasta nueva si el servidor continúa ofreciéndolas.

Este sistema está pensado para funcionar con un número máximo de clientes, en este caso se ha establecido a 10 y no serán aceptados más clientes a partir de ese número de conexiones. Una vez hayan entrado los clientes a la primera subasta, se cierra la posibilidad de que nuevos clientes entren a subastas posteriores, ofreciéndolas únicamente a los que accedieron a la primera subasta.

Procesos de gestión

Antes de que se creen las conexiones y se lancen los procesos que servirán a cada cliente, una serie de procesos se encargará de realizar diversas tareas administrativas de forma concurrente:

Estadístico

Periódicamente se muestra por pantalla información estadística del sistema: número de imágenes mostradas, tiempo total y medio que las imágenes han sido mostradas, número de peticiones encoladas y tiempo contratado estimado.

Para ello, mientras no ha comenzado la terminación ordenada de las subastas el proceso se duerme durante un tiempo determinado,

posteriormente llama a una función del monitor de la que recibe un string con la información estadística y llama a otra función del monitor que muestra la cadena por pantalla en exclusión mutua.

Administrador

El proceso se duerme durante un tiempo determinado. Una vez termina ese tiempo notifica a todos los procesos servidor que se debe proceder a la terminación ordenada de las subastas, de manera que el servidor espera a que termine la subasta en curso, pero no se realizarán más subastas (esto no afecta al gestor de vallas, que seguirá ejecutándose hasta procesar todas las imágenes encoladas).

Estas informaciones se notifican al resto de procesos mediante funciones del monitor para preservar una concurrencia efectiva.

Gestor de vallas

Mientras se estén realizando subastas, procesa las peticiones que se encuentran en cola para colocar anuncios en vallas publicitarias. Al procesar una petición, muestra durante un tiempo determinado en una ventana la imagen que se encuentra en la URL proporcionada por el cliente. Si el tiempo de subastas ya ha acabado, pero se encuentran peticiones en cola, continúa ejecutándose hasta procesar todas ellas.

Su funcionamiento se basa en el código proporcionado como ejemplo, y se ha modificado levemente su estructura. Las vallas se lanzan con un formato determinado para se asemeje más a las vallas publicitarias reales.

Conexión

Para establecer la conexión entre el servidor y los múltiples clientes, previamente se lanza un proceso que se duerme durante un tiempo prefijado, el cual será el tiempo del que dispondrán los clientes desde que se inicia el servidor para que puedan “entrar a la sala de subastas” y así comenzar las sucesivas subastas en cuanto este tiempo venza.

Se ha tomado la decisión de no permitir nuevos clientes para próximas subastas si no acuden al primer llamamiento, para simular un entorno más realista de subasta privada, en el que los clientes tienen que acudir a una hora determinada si quieren continuar. El cliente que no llegue a la hora acordada no tendrá derecho a participar en las subastas, de esta forma solo aceptamos clientes puntuales dispuestos a participar en una o más subastas.

El modelo utilizado es muy similar al utilizado en la práctica 5, donde para atender a cada cliente se lanzan sucesivos procesos dentro del servidor.

La aceptación de cada uno de los clientes por cada proceso se realiza mediante una instrucción “accept” bloqueante que esperará hasta poder obtener un descriptor del cliente válido.

Complementariamente se lanza un thread para que en el caso de que se quede bloqueado el servidor en el “accept” porque no han entrado el número máximo de clientes permitidos, se crea un cliente virtual que por enlace local se conecta al servidor para desbloquear la instrucción bloqueante y se permita volver a tomar el control del thread principal del servidor.

Una vez se haya creado y aceptado la conexión con el cliente y se haya lanzado el thread correspondiente, se incrementa en una unidad el número de pujadores.

Subasta

Estructura

Para simular la subasta nos servimos de una estructura que contiene los siguientes campos:

Un precio de salida aleatorio entre 100 y 1000 €, un tiempo (segundos que se mostrará la imagen) equivalentes a una vigésima parte del precio de salida, un precio de reserva 50% mayor que el de salida, un precio de incremento equivalente a un 66% del precio de salida, y un precio actual correspondiente al precio de salida.

Funcionamiento

El protocolo de subasta empleado se corresponde con la interpretación que ofrece la FIPA sobre la ejecución de una subasta inglesa.

El proceso que sirve al cliente comienza la subasta cuando se le notifica que debe continuar la ejecución del código mediante el thread que lanzamos en el apartado de conexión, usando el monitor para una correcta sincronización. De esta forma los clientes empezaran la ronda de subastas al mismo tiempo.

Este proceso consiste en un bucle que se ejecuta mientras no se notifica el fin de las subastas y el cliente no se ha desconectado.

Se envía al cliente un mensaje: *"START"*, que le notifica el inicio de la subasta; a continuación, entra en un segundo bucle que se ejecuta mientras el cliente no se ha desconectado y el cliente no haya terminado de participar en esa subasta.

Se envía un mensaje al cliente que contiene el precio actual de la subasta y el servidor recibe una respuesta por parte del cliente que le indica si está dispuesto a pagar el precio propuesto: *"ACCEPT"*, por lo tanto, continua la subasta en posteriores rondas o le dice que rechaza el precio: *"REJECT"* de forma que esperará a que el resto de pujadores terminen la subasta.

El sistema de subastas se ha planteado atendiendo a criterios de igualdad en cuanto a clientes y una mejor sincronización entre los propios clientes y el servidor.

De esta manera se espera a que todos los clientes hayan aceptado o rechazado la propuesta (se entiende como rechazado una respuesta incorrecta o una desconexión inesperada) para después analizar esos resultados e incrementar el precio en el caso de que haya dos o más clientes interesados en la subasta.

Tras recibir este mensaje, el servidor envía un nuevo mensaje al cliente en forma de entero informándole de su situación:

```
0: Ha rechazado la propuesta.

1: Ha decidido aceptar la puja, pero aún no ha ganado.

2: Ha aceptado la puja y además ha ganado, pero no llega
al precio de reserva.

3: Ha aceptado, ha ganado y llega al precio de reserva.
```

Si el cliente que ha aceptado la puja era el único cliente que se encontraba en la subasta y el precio que ha aceptado supera el precio de reserva, el cliente le proporcionará al servidor la ubicación web de la imagen publicitaria.

El servidor encolará una estructura de datos de la valla a mostrar, esta estructura es de tipo “datosValla”, la cual está compuesta de:

Una dirección URL dónde se encontrará la imagen a mostrar, un entero que determina el tiempo que se va a mostrar el anuncio, una cadena de caracteres que servirá para identificar que cliente compra el anuncio y un entero con el precio por el que se ha vendido el anuncio.

Monitor

El monitor controla todos los aspectos dedicados a la concurrencia en el servidor, nos permitirá aislar las secciones críticas y conseguir comportamientos como por ejemplo que solo un proceso ejecute determinadas instrucciones. Dado que el monitor realiza muchas acciones y muy diversas e independientes entre sí, se han utilizado múltiples mutex:

```
recursive_mutex finMtx: Lo utilizamos para controlar
el final de todas las subastas.
recursive_mutex colaMtx: Se emplea para todas las
operaciones referentes a la cola. Además, se utiliza
una condición variable para dejar en espera el gestor
de vallas siempre y cuando no haya datos en la cola.
recursive_mutex pujadoresMtx: Es necesario para la
conexión y desconexión de participantes, así como para
```

contabilizar clientes que aceptan y rechazan en cada ronda.
recursive_mutex inscripcionMtx: *Es únicamente utilizado junto con una variable condición para mantener en espera a los clientes mientras se encuentra el servidor en proceso de inscripción.*
recursive_mutex textoMtx*: Es usado para mostrar texto por la salida estándar en exclusión mutua para evitar entrelazados de mensajes.*